



Universiteit
Leiden
The Netherlands

Opleiding Informatica

Implementing a User Interface for Attack-Defense Tree WebApp
Using Human-Computer Interaction Principles

Marwa Mohalaia

Supervisors:

Mr. N. D. Schiele & Dr. O. Gadyatskaya

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)

www.liacs.leidenuniv.nl

23/06/2023

Abstract

The applications of the security modeling tool, *attack-defense tree*, are not extensively utilized. One of the underlying causes is the design flaws in the interface of these applications. This research studies how to provide a practical solution for this issue by using *human-computer interaction* principles. Consequently, both theoretical and practical considerations are covered to apply the principles on the interface of the *attack-defense tree* tool. The ultimate goal of this study is to build a user-friendly interface for a web application that allows users to create and customize *attack-defense trees* without the demand for technical knowledge. Therefore, the results provided are the final interfaces of the web application, followed by a usability-based evaluation. Finally, the project limitations are addressed along with future work ideas to optimize and complete the achieved results.

Contents

1	Introduction	1
1.1	Research Objectives	1
1.2	Thesis overview	2
2	Background	3
2.1	Attack-Defense Tree (ADT)	3
2.2	Human Computer Interaction (HCI)	4
3	Related Work	5
4	Methodology	7
4.1	Theoretical Framework	7
4.1.1	Requirements	7
4.1.2	WebApp Structure	9
4.1.3	HCI Principles	10
4.1.4	Visual Prototyping	13
4.2	Technical Implementation	15
4.2.1	Choice of Technologies	15
4.2.2	Front-End Development	15
4.3	Heuristic Evaluation	18
5	Results	24
5.1	Interfaces	24
6	Evaluation	27
6.1	Comparison	27
6.2	Limitations	28
7	Conclusion	29
7.1	Future Work	29
	References	33

1 Introduction

Most digital systems have vulnerabilities in their security that attract cybersecurity criminals and attackers. Reports in recent years have consistently demonstrated an alarming increase in security breaches and the consequential monetary losses resulting from cybercrimes. A notable study conducted by IBM Security in 2022 [1] examined data breaches in 550 organizations, revealing that 83% of these organizations experienced multiple data breaches within a year. Additionally, the average total cost of a data breach was reported to be 4.35 million USD. These findings emphasize the escalating severity of the situation. Consequently, it becomes crucial to prioritize robust system security. One effective method in this regard is the *attack-defense tree*.

Attack-defense tree (ADT) tools are used by a range of cybersecurity professionals, IT teams, risk management teams, and systems managers to help them evaluate the security of systems and develop effective defense strategies by modeling the scenarios of attacks and the actions that should be taken as defense [2]. These tools provide organizations with a clear understanding of the sequential nature of cyber attacks by identifying vulnerable entry points and corresponding countermeasures for each scenario. However, although ADT is highly valued in academia for its benefits and advantages, its adoption in practice remains low, which causes a significant gap between academia and the industry to exist. One of the keys to bridge this gap is to increase the usability of ADT by providing a user-friendly tool for non-technical practitioners, who need to assess the security of their systems with ADT.

Usability plays an essential role in security tools by directly impacting user satisfaction and technology adoption. Moreover, complex and non-user-friendly tools have the potential to discourage users from using them and cause usage errors. Therefore, prioritizing usability is important to ensure that users are satisfied and can easily understand security tools, as demonstrated in a survey conducted by Katsabas et al. [3]. *Human-computer interaction* (HCI) principles and user experience aspects help improve the interfaces of tools and applications by optimizing the interaction method between users and computers. These principles have demonstrated significant improvements in the usability of applications, specifically interactive systems [4]. Moreover, HCI has effectively transformed these tools and systems, making them more feasible and robust, particularly from the users' perspective. Therefore, using these design principles in the built tools will help increase the usability of ADT and allow practitioners to use this technique easily.

1.1 Research Objectives

This research aims to build a user interface for an *attack-defense tree* web-based application by taking into account the principles of *human-computer interaction*. Therefore, the main goal is to accomplish the following objectives:

1. **O1:** Identify the HCI aspects that should be considered in applications' interfaces to increase their usability.
2. **O2:** Build a user interface of an ADT web application for non-technical practitioners by applying HCI principles.

1.2 Thesis overview

This chapter 1 contains the introduction of the research with the research objectives; Chapter 2 includes the definitions of *attack-defense tree* and *human-computer interaction* principles; Chapter 3 discusses some previous work of similar research on ADT and HCI; Chapter 4 describes the methods and the implementation of the ADT application interface, including the theoretical part as well as the technical part, which ends with an evaluation; Chapter 5 demonstrates the final results of the web application interface design; Chapter 6 discusses the findings and addresses a comparison with previous work, besides, it summarizes the limitations; And finally, Chapter 7 draws the conclusion of this study and addresses future work ideas. This thesis plan is visually summarized in Figure 1.

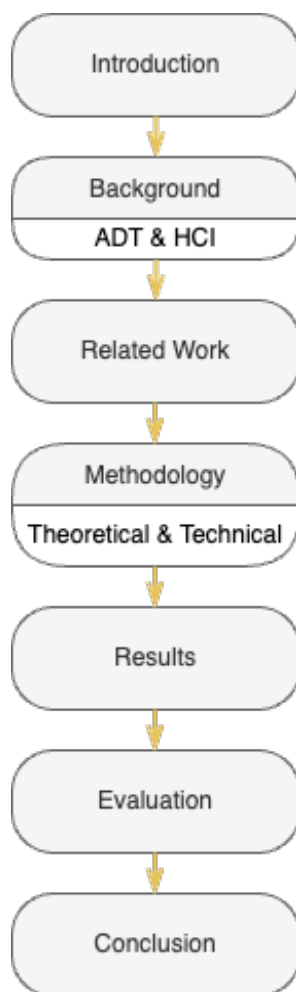


Figure 1: Thesis chapters overview

2 Background

2.1 Attack-Defense Tree (ADT)

Attack-defense tree is a security modeling tool that helps evaluate the security of systems by representing the scenario of the attack [2]. The tree consists of labeled nodes and refinement components. The labeled nodes describe the scenario of the attack, including the defense methods. Therefore, nodes can be either attack nodes or defense nodes. Each node has a refinement component to specify the action of its child nodes. If the refinement of a node is of type “AND”, then all of its children should be achieved. Whereas if the refinement is of type “OR”, then one or more child nodes can be achieved [5]. In the following example, the representation of each component can be seen in Figure 2.

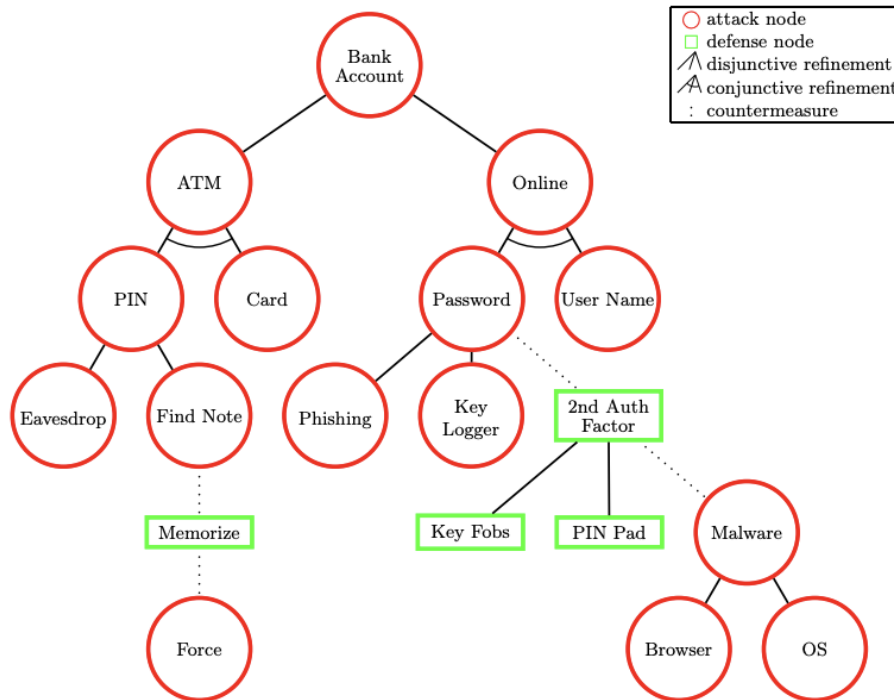


Figure 2: Attack-defense tree example [5]

Figure 2 presents one of the standard visual notations utilized in ADTs, introduced by Kordy et al. in [5]. The provided diagram offers an illustrative instance of a bank account attack scenario. The root node represents the primary objective of the attack, while the red circles illustrate the attack nodes. Correspondingly, the green rectangles represent the defense nodes associated with the parent (attack) node by a dotted line, such as the one between the “Memorize” node and the “Find Note” node in the example. Moreover, the arc between two branches emanating from a node, such as the arc in node “Online” and “ATM”, signifies an “AND” refinement, denoting that both conditions specified in the child nodes of the node with the arc are required for the successful execution of the attack. Conversely, the absence of such an arc represents an “OR” refinement, indicating that either condition can be satisfied to progress further.

2.2 Human Computer Interaction (HCI)

The design of computer technologies is essential for their success as well-designed computer applications significantly enhance performance, efficiency, and usability. This leads to increased user interest and preference over other similar applications, which contributes to the success of these preferred well-designed applications. A key tool for implementing a more efficient design for computer technologies is the design field of *human-computer interaction*.

Human-computer interaction (HCI) is a field that studies the design of computer technologies which are based on the interaction between humans and computers. HCI focuses on improving the design of computer technologies by providing them in a user-friendly environment that meets the expectations and the requirements of users. As a result, when users interact with these applications, they will have a satisfying experience, that will allow them to use and access the applications consistently, without complications. Consequently, this will help improving the usability of the applications and accomplishing high error-free performance. However, HCI studies multiple types of technology design, such as software system development, hardware system design and application interfaces. In this research, HCI techniques for designing user interfaces will be studied and applied on a web-based application.

HCI has several principles that are applied in the design process. The most important concepts are the eight golden principles by Schneiderman in his book “Designing the user interface” [6] and the seven principles by Norman in his book “The design of everything” [7]. These principles overlap and address similar areas. They can be classified into three categories [8]: visual presentation (Visibility), user communication with the application (Interaction), and user experience (Usability), as illustrated in Figure 3.

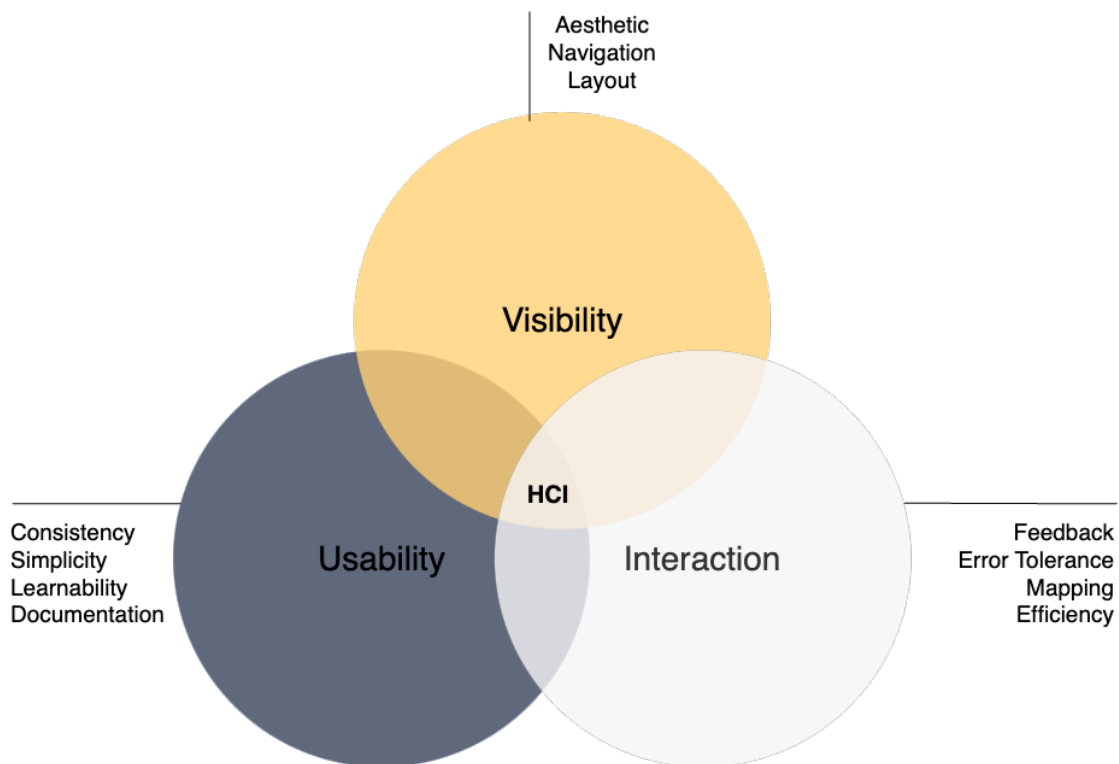


Figure 3: The principles of interface design in HCI

3 Related Work

This chapter provides an overview of the relevant research carried out in the same area of the current research. This encompasses various aspects of *attack-defense tree and human-computer interaction* principles. However, to the best of my knowledge and the research conducted for this study, the issue of user interface usability, particularly for the *attack-defense tree* applications, has not been previously addressed in the literature. The two areas have been discussed independently in numerous studies. Consequently, we will first refer to the research on the *attack-defense tree* and then review the research that has examined usability and HCI principles.

Delcourt and Weiser [9] delved deeply into ADT and provided a complete analysis of its features and the tool developed by them. The tool has a simple user interface with a tree display window, a reload button and a file name field. Besides, it provides a text window that has tree grammar space, an import button, create button and a JSON file name field. However, the tool delivers more in-depth features that interact with the data of the tree and output solutions for the scenarios of the attacks. For example, it allows the user to compare two ADTs and generates random trees. Likewise, Opel [10] has designed and implemented a supporting tool for generating ADT with various functionalities.

A well-known tool for generating and evaluating *attack-defense trees* is ADTool [11]. It is a free open-source software that provides graphical and quantitative analysis for security attack scenarios. The software's features were elaborated by Kordy et al. in [12]. Although the tool was not designed based on user experience criteria, it provides a simple interface with evaluation methods for calculating the costs of attack scenarios and showing the worst-case scenario. However, a more advanced version known as ADTool2.0 was introduced by Gadyatskaya et al. [13]. This tool provides a wide range of functionalities for attack trees, attack-defense trees, and attack trees with the SAND operator [14]. Moreover, it offers advanced features for visualizing, scripting, and analyzing attack scenarios.

In a similar research to the studies of ADT modeling, Ruijters and Stoelinga [15] have explained in detail the aspects and features of fault-tree. Fault-tree shares similar fundamentals with the ADT, but places greater emphasis on evaluating safety-related risks. The methodology involves constructing a tree structure that identifies combinations of events or failures that could result in an undesirable outcome. Each cause within the main event is further examined and expanded into multiple sub-causes. The paper presents a thorough investigation into modeling techniques, probability calculation methods, and diverse approaches through practical examples.

Pinchinat et al. [16] introduced ATSyRA, a tool developed with a focus on enhancing the physical security of military buildings. This tool automates the process of generating attack trees for system analysis by providing editors that enable users to describe a system and create attack trees using advanced actions. Additionally, it incorporates interactive synthesis to facilitate the refinement of the attack tree in a dynamic and interactive manner.

Kordy et al. [14] have also introduced SPTool, a tool that incorporates the sequential conjunctive operator (SAND) in SAND attack trees. This tool is an equivalence checker that enables the distinction between actions that must be executed sequentially and those that can be performed in parallel. Similarly, Kumar et al. [17] presented ATTop tool for analysing attack trees using a model-driven engineering approach.

Lallie et al. [18] addressed the variations in the visual syntax of cyber-attacks in attack trees. The authors focused on analysing the different methods used in modeling cyber-attack scenarios

and emphasized the importance of having them unified in a standard method.

Several tools have been developed for ADT [19],[20],[21] with a main focus on functionalities rather than user interface design. These tools offer valuable features for creating and analyzing ADT models, with prioritizing security analysis services over user interface improvements.

Hong et al. [22] have performed a survey on various graphical security models (GrSM) for network security systems, including ADT [5], PT [23], AG [24], LAG [25], EDG [26], BAG [27], CG [28], TVA [29], SAG [30], CMG [31], HARM [32] and other models, to enable users to distinguish the best method to use according to their security needs.

Moving to the domain of HCI research, Gong et al. [8] studied the principles of HCI closely and analyzed them in depth. Furthermore, the authors categorised the principles into the three groups, which are used in the current research, based on user experience study. For the purpose of creating a systematic software interface design model, they provided design strategies and used the optimization theory of “three transformation and fusion” of software interface in order to improve user satisfaction.

Polasanapalli and Buggareddy [4] conducted a study on the design principles of HCI and devoted their attention to an interactive system. They analyzed the impact of design changes based on HCI concepts and user-based surveys. The findings demonstrated the effectiveness of UI design concepts in enhancing the usability of interactive applications.

Kandababu and Indukuri [33] carried out a complete analysis of usability and HCI principles on the user interface of an internet booking system. Their research focused on studying the design aspects of the Ryanair.com website, identifying usability flaws, and evaluating user experience and usability using heuristic evaluation methods.

Adhitya et al. [34] performed research on a web-based application that employs a similar methodology to the current study. However, their main emphasis was on applying the User-Centered Design (UCD) approach to analyze the user interface and the user experience of the website under investigation.

Dix et al. have presented a comprehensive collection of concepts related to HCI in their book “Human-Computer Interaction” [35]. This includes the foundations, design process, models, and theories. The book serves as a valuable guide for HCI by summarizing all the essential aspects of design. Similarly, Rogers et al. [36] have extensively discussed interaction design aspects. Norman [7] has previously addressed the design aspects as well by exploring the relationship between design and human psychology. They provided insights into the ways in which design impacts human behavior and cognition.

Zubrycki et al. [37] introduced Robokol, a user-friendly graphical interface designed to simplify robot programming for non-practitioners, with a particular focus on autism therapy. Their emphasis was on creating an intuitive interface that can be easily utilized without requiring extensive technical knowledge.

Hinze-Hoare [38] investigated the overlapping principles of HCI, recognizing that there is no consensus on a single set of HCI principles. Instead, multiple resources exist that group them differently, such as the eight golden principles proposed by Schneiderman [6] and the seven principles by Norman [7]. To identify the most widely recognized HCI principles, Hinze-Hoare conducted an analysis of usability criteria based on the frequency of author citations, resulting in the identification of eight key principles for HCI.

4 Methodology

In this chapter, the process of implementing the objectives (1.1) of the research will be discussed, following the steps shown in Figure 4.

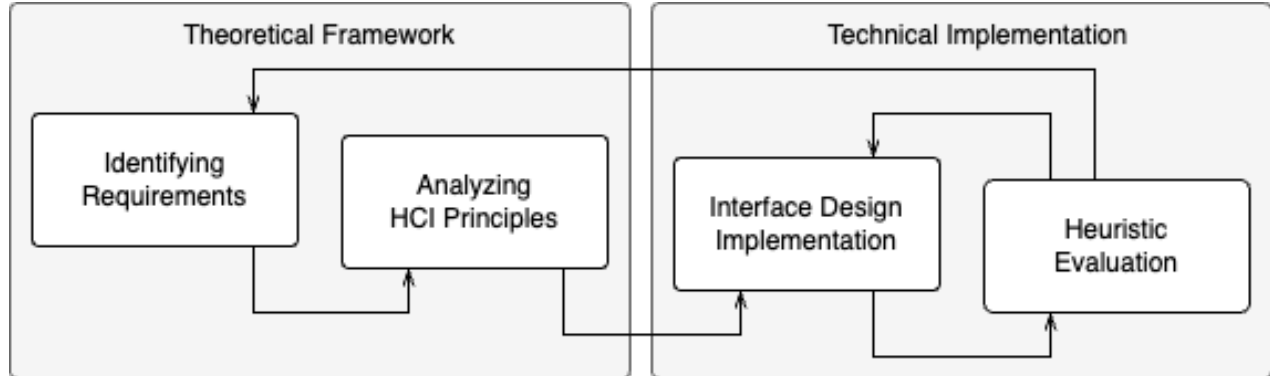


Figure 4: Research method flow

In the implementation phase, we will first study the requirements of the ADT tool, then will work on the first objective 1, which includes identifying the essential aspects of interface design by analyzing the HCI concepts and the procedure for implementing these concepts in the web application. Secondly, the theoretical implementation of the second objective 2 will be explained. This includes building the web application based on the phases and principles followed, along with the technical implementation which will be done using the relevant web design tools. Finally, the results are discussed and a usability heuristic evaluation is performed on the web application.

4.1 Theoretical Framework

4.1.1 Requirements

Target Group

The purpose of the ADT web application is to provide a platform for diverse security system users to help them simplify the process of assessing security vulnerabilities and allow users to effectively analyze potential attacks with ADT without requiring a high level of technical expertise. The scope of using the tool is not limited to programmers or developers of security systems, but also includes students, regulators, business analysts, managers, and others who need to have an analysis of a relevant system. Therefore, the target users are non-technical practitioners who need to use the ADT tool to analyze security attacks easily.

Use Cases

In order to provide users with the best experience when using the ADT design tool, it is necessary for the tool interface to perform several functions that align with the specifications of ADT, as well as meet the expectations of the user. Based on this, the web application needs to implement the use cases illustrated in Figure 5.

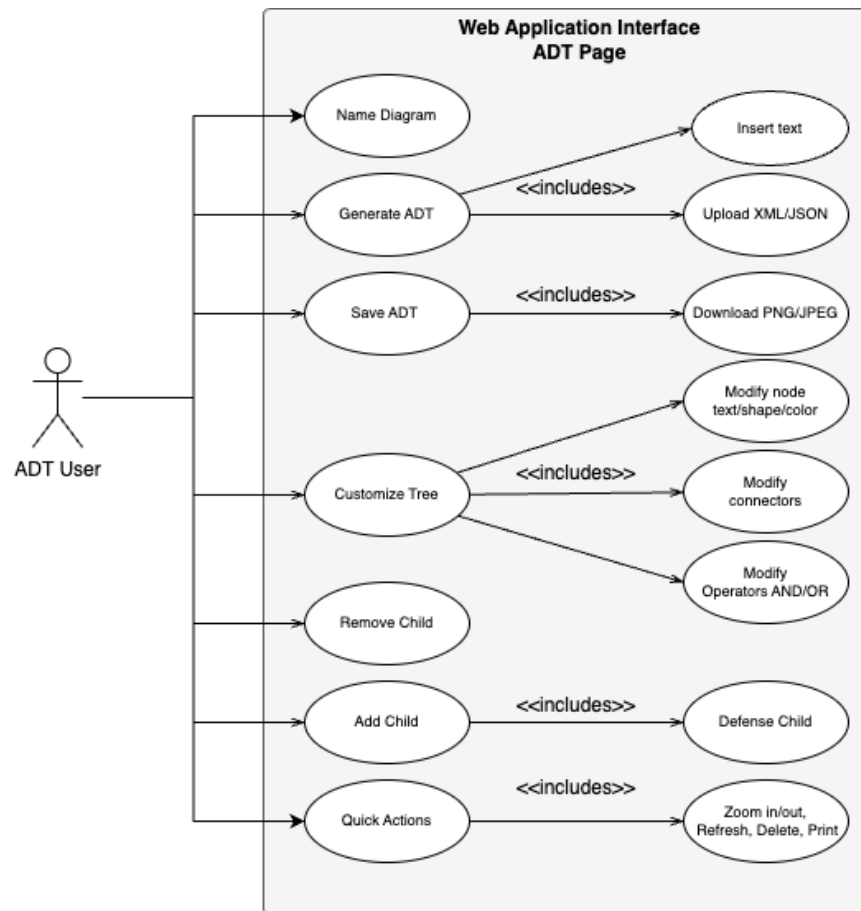


Figure 5: ADT web application use case diagram.

In close collaboration with my supervisor, Mr. Schiele, we worked together to define the functions and the use cases needed for the web application, based on the user requirements and the ADT features. The user of the tool requires the ability to interact with the interface and perform the functions provided in Figure 5 as follows:

- Generate ADT
 - Generating ADT by inserting a text (using simple markup language)
 - Generating ADT by uploading XML or JSON file that describes the tree.
- Add Child
 - Add the text of the child node.

- Mark the new child as defense child.
- Remove child.
- Customize ADT
 - Change node’s text.
 - Change node’s shape.
 - Change node’s color.
 - Change operators (AND/OR).
 - Change connectors (connected/dotted) lines.
- Save ADT
 - Export the ADT diagram and download it as PNG or JPEG.
- Quick actions
 - Zoom in/out
 - Delete tree
 - Refresh/reset diagram
- Name ADT diagram.

4.1.2 WebApp Structure

Based on the use case requirements, the tool demands having the pages illustrated in Figure 6:

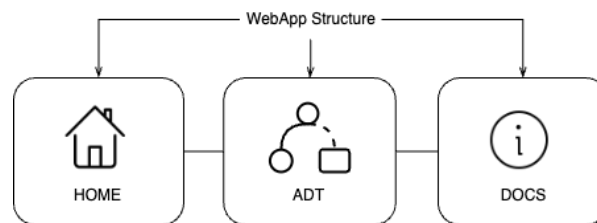


Figure 6: ADT web application structure.

We can navigate to the three pages structured in Figure 6 by using the navigation bar menu. Each page can be accessed via the menu bar from any other page. Below, a description of each page is provided:

1. HOME page
 - A page where the tool and its features are introduced to the users.
2. ADT page (main page)
 - A page where the tools and the visualization of the ADT are used.
3. DOCS page
 - A page where the instructions and the documentation for using the tool are provided.

4.1.3 HCI Principles

In the field of HCI, a range of principles have been identified by Schneiderman [6], Norman [7] and others [35][36], as discussed in chapters 2.2 and 3. To address the first objective of our research, we have selected eleven web-design-related principles from the mentioned resources to be studied and applied to the ADT web application. Moreover, we grouped the principles using the three categories classified by Gong et al. [8], which are: visibility, usability, and interaction. To ensure the effective implementation of these principles in the front-end of the ADT tool, we will analyze each principle and determine its specific implementation requirements in the tool. These requirements are based on the target audience and the use cases discussed previously in 4.1.1.

1. Visibility Principles

Navigation: The navigation system enables the user to determine his location within the website or the system, or basically, it represents the structure of the website. In our tool, the web application has three pages to which the user can navigate. Another navigation element in the tool is the sidebar or “Tools” tabs [36].

- *Implementation requirement(s):* The ADT web application has a clear navigation bar and a sidebar with clear labels for each tab.

Aesthetic: The design should be aesthetically pleasing and attractive to users. This includes using balanced colors, clear visual components, high-quality images and well-designed items[39].

- *Implementation requirement(s):* The ADT web application uses simple colors (navy blue, white and gray for objects’ backgrounds and yellow for highlighting). It also has attractive high-quality icons, and smooth animated transitions when moving from one tab to another in the menu, as well as when hovering over the items.

Layout: The layout refers to how the visual elements of a web page are organized. This includes grouping items that belong together, listing related items in a clear structure, ordering particular items when necessary, choosing colors that help with splitting white spaces, and finally having a clear alignment for the objects and texts. A well-designed layout reduces cognitive load and helps users understand the elements easily [36].

- *Implementation requirement(s):* The ADT web application has a clear layout where the tools are grouped and ordered according to their functions in the left sidebar. Further, the tree visualization area is clear and has the largest part of the layout. In addition, diverse contrasting colors are used to highlight the scopes of different page components, with spaces between them.

2. Usability Principles

Consistency: Consistency refers to maintaining the same attributes and functions of similar design elements across the interface, including colors, shapes, features, input and display methods. Besides, attempting to bring the performance experience closer to the real-world experience, with considering delivering familiar behaviors to the user [7].

- *Implementation requirement(s)*: The ADT web application uses identical shapes and colors for input forms (textbox with a button) and buttons (yellow, rectangle), beside similar interaction methods with similar elements. In addition, the tool provides the user with a familiar experience, for example, the motion of opening and closing the tabs is expected to open the hidden part of each tab, and the action of pressing the buttons is directly triggered by a click. Moreover, the function of the clicked icons performs as indicated in the icons/buttons, for example, the button with the trash can icon is expected to delete the diagram.

Simplicity: The interface elements should be simple to access and use to ensure that users do not experience frustration when accessing a particular element or when performing a specific task[36].

- *Implementation requirement(s)*: The ADT web application uses understandable elements with a clear appearance. This is achieved by providing simple text and icons. Furthermore, users are able to generate the tree and customize it with a single click on the option of the desired adjustment. For example, the process of changing the node color requires selecting the target node and a click on the target color in the menu. A likewise simplified approach is applied to the rest of the available functions of the tool.

Learnability: Learnability is the ability of users to learn how to utilize the tool effectively in order to perform tasks independently. This principle is closely associated with the rules of consistency and simplicity, whereby the combination of these three principles enables the users to learn progressively, which leads to the principle of **memorability** [6]. Consequently, users are able to remember what they have learned by interacting with the application interface [36].

- *Implementation requirement(s)*: The ADT web application provides clear usage instructions for the user to learn how to use the tool. For example, a placeholder text is used as an indication of what the user should enter in the text boxes in use. Once the user learns the input format the first time, he will be able to enter the required format directly in the second or third time.

Documentation: This feature helps users understand the application and its use, it includes the use instructions, frequently asked questions and possible errors. The presence of this feature is important for users as they may encounter difficulties in performing certain tasks or may be prevented from completing them [39].

- *Implementation requirement(s)*: There is a separate page created with instructions for the user on how to use the tool and the errors that may be encountered. The use of this feature by clicking on its page name in the navigation bar should not prevent the user from completing the tree generation process and should not lead to loss of the data they have already entered or obtained. This is avoided by launching the instructions page in a new tab.

3. Interaction Principles

Feedback: Feedback is displaying the effect of the action taken by the user to be informed whether the task performed was successfully completed or failed. Feedback is important because it provides the user with information about the current state of the application. It can be implemented through audio, visual or textual elements [7].

- *Implementation requirement(s):* The tool directly displays to the users the modifications they made on the tree nodes. In the case of form submission, a text message will be displayed to inform the user of the completion or failure of the process.

Error Tolerance: Refers to the ability to recover from an error that occurs while performing a task, to ensure that the desired action can still be achieved. Users are likely to make mistakes when working with applications, especially during their first time. Therefore, they must be able to handle any error, either through Forward Error Recovery, which indicates the ability to continue the task by performing the correct action from the current failed state, or through Backward Error Recovery, which indicates the ability to return to the state before the error occurred [7].

- *Implementation requirement(s):* The ADT web application uses constraints on the input fields, for example, users can not submit empty forms or customize non-selected nodes. Furthermore, users can delete a single child node or refresh the diagram to return to the previous state. Moreover, the configurations provided allow users to switch between any selected option, for instance, if the refiner operator AND is mistakenly clicked, the operator OR can easily be chosen back. The same applies to colors, shapes and connectors.

Mapping: This principle refers to mapping the controls of the interface with their functions. Hence, the design of the interface components should represent their functionality and what the user is expecting from an element should be performed as intended [7].

- *Implementation requirement(s):* The interface buttons function exactly as they represent. For instance, the “Zoom”, “Delete”, “Refresh”, “Download” and “Upload” buttons deliver the expected tasks. Additionally, submitting forms and customizing the tree are visually perceived as expected and provided by the corresponding interface objects.

Efficiency: Efficiency focuses on reducing the time spent by users on completing a task. This objective can be achieved by combining the principles of Simplicity, Learnability, Feedback and Error Tolerance together on the design [36].

- *Implementation requirement(s):* The ADT web application provides efficient methods to complete each task. For instance, if a user intends to add a new child node to the tree, they can easily locate and open the related “Child” tab. Once they select the target node, they will see the option to add a new child node through simple and clear natural language. Additionally, modifying a selected node can be accomplished with just a few simple actions.

4.1.4 Visual Prototyping

The visual representations of the web application interface design are designed following three stages: Wireframe Design, Mock-up Design and Prototype Design. The interface design decisions are made based on the addressed target group, the listed requirements of the tool and the discussed HCI principles.

Wireframe

In Figure 7 the basic layout of the tool page is illustrated.

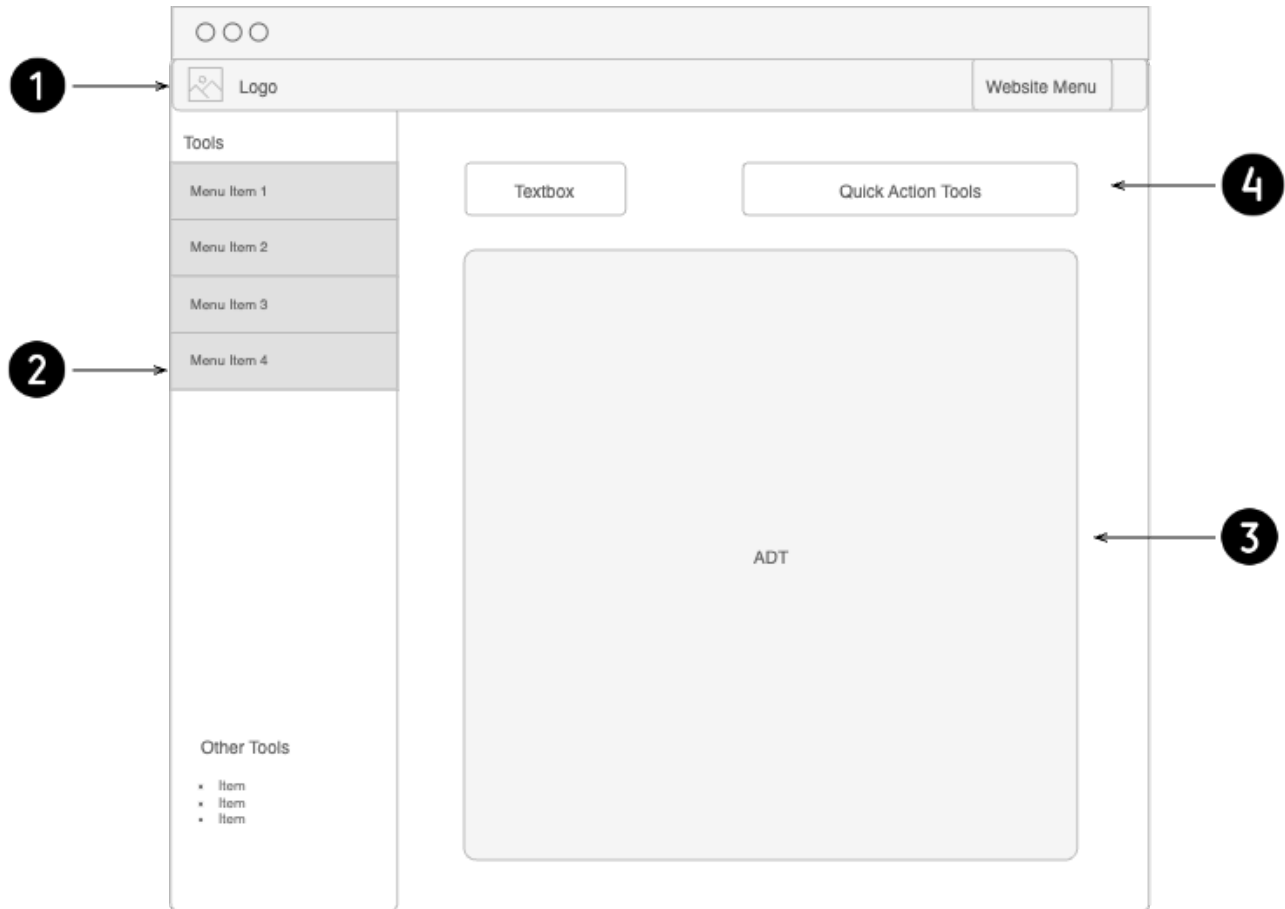


Figure 7: ADT web application wireframe design

In the wireframe design 7, we can distinguish the main elements of the application and the scope of each. The page features a navigation bar at the top 1, a side menu on the left that contains most of the important tools 2, a large area for visualizing the tree 3, and a section for simple tools at the top of the tree section 4.

Mock-up

In order to examine the design details of each element of the interface, a Mock-up version is created to incorporate various aspects of the application's tasks. This stage enables us to observe the development of the final tool model, which is highly dependent on the mock-up design in Figure 8.

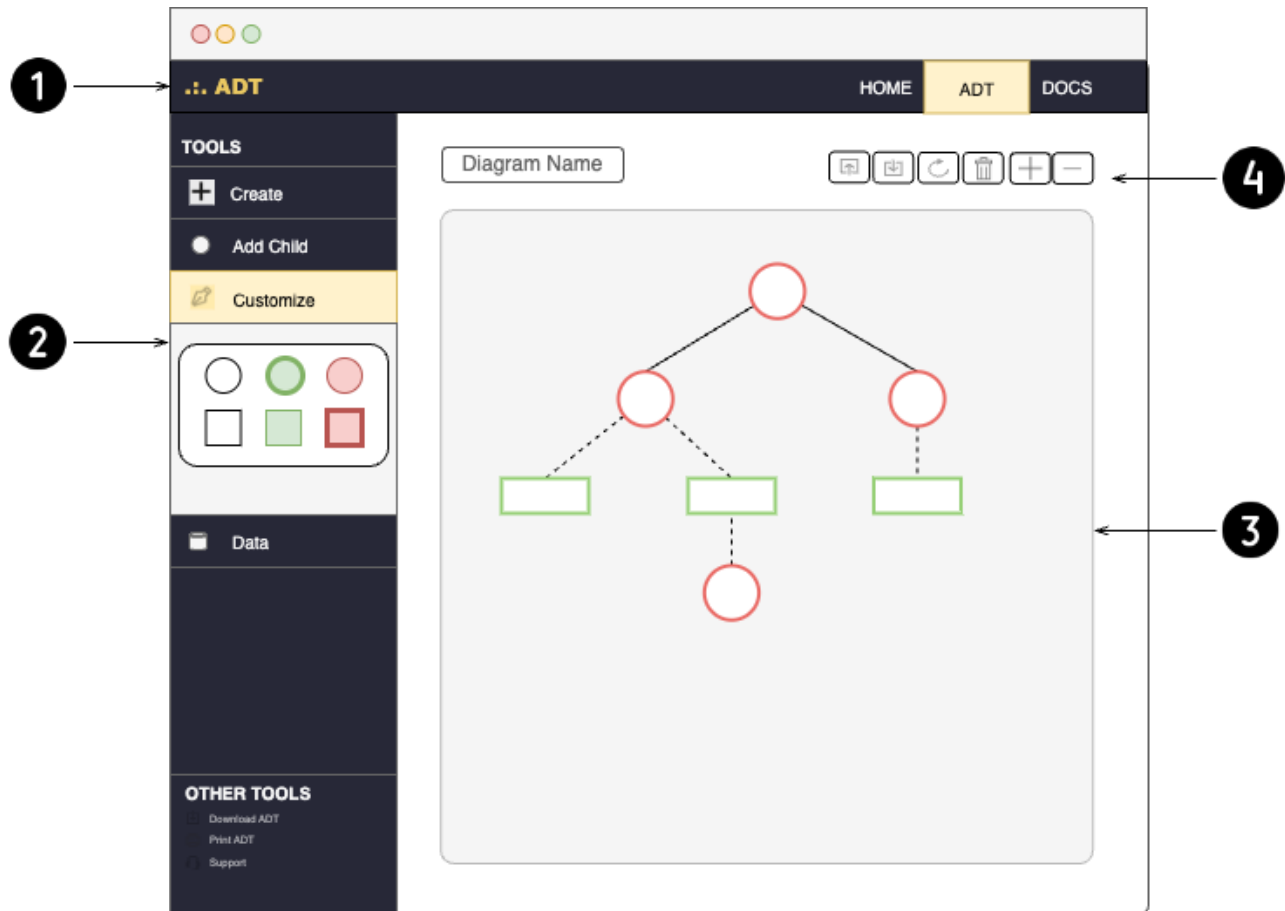


Figure 8: ADT web application mock-up design

In this version 8, more colors, shapes, icons and texts are added. This includes the logo and pages names ①, the tabs names, tools and icons on the sidebar ②, a basic visualization of the tree ③, and finally more detailed quick action buttons ④.

Prototype

The prototype design introduces the final interactive version of the ADT web application. This version is similar to the technically developed version, which will be elaborated upon in the section regarding technical implementation (see 4.2) and in the final results (see 5.1).

4.2 Technical Implementation

4.2.1 Choice of Technologies

To design the interfaces of the application, multiple design and programming languages are used.

- **HTML:** (Hypertext Markup Language) Is used to create the elements of the web page.
- **CSS:** (Cascading Style Sheets) Is used to modify the style of the HTML elements.
- **JavaScript:** Is used to add functions to the HTML elements and control them.
- **Bootstrap v4.0:** Is an open-source CSS framework. Its free templates of website components were used and modified across the pages.
- **P5.js:** Is a JavaScript library that provides an easy method to develop interactive visual elements. It is used to connect the front-end interface with the back-end code.

4.2.2 Front-End Development

After analyzing the web application requirements, studying HCI concepts, and designing the prototypes, we will now discuss the design decisions and their practical application. The front-end design was initiated by creating the `index.html` page and adding the required CSS, JavaScript and Bootstrap libraries. After designing the page structure, Bootstrap blocks were used for each part of the page layout, with the aim of modifying them later, as shown in Figure 8. Accordingly, CSS and JavaScript files were created to adjust the page elements by adding the related code.

Block: Navigation Bar

This part (Figure 9) has the tool's logo with the name of the tool on the left, and the list of pages on the right (Home, ADT, and Docs). The black color of the navbar was chosen to distinguish it from the rest of the page. Conversely, the white color of the text was chosen for font contrast and readability. To ensure ease of access, the page names are positioned on the right-hand side of the screen.



Figure 9: The navigation bar of The ADT web application.

Block: SideBar

The black color of the sidebar was chosen to distinguish it from the largest area of the page, which contains the tree. Similarly, the white color of the tab names was chosen for clarity. To indicate the user's current location within the menu, the background of the menu being hovered over is highlighted in yellow. Additionally, when a menu is opened, the menu icon color changes to yellow, providing a visual distinction between the opened and closed tabs of the sidebar menu. Finally, a small line was added between the list of primary and secondary tools to make it easier to differentiate between them. However, The sidebar (Figure 10) consists of the following tabs:

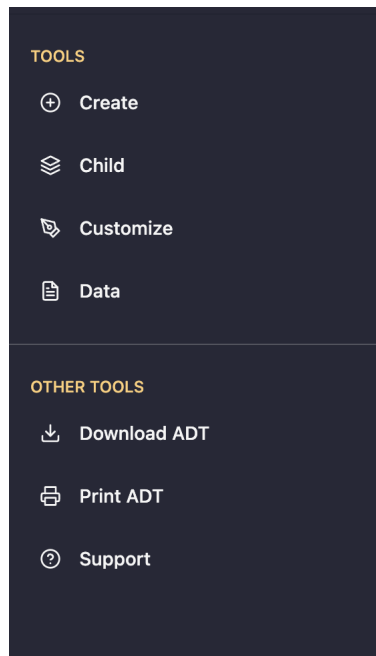


Figure 10: The sidebar of The ADT web application.

- **Tab: Create**

This panel has a `<textarea>` for the plain text input of the tree with a placeholder, and a `<button>` to submit the input.

- **Tab: Child**

This panel has two parts, one for adding a child, and one for removing a child. To add a child, the user should enter the contents of the added node in the `<input>` tag, and choose whether it should be created as a defense node or an attack node. Similarly, to remove a node, the user should select it or enter its text in the `<input>` of removing a child and click the removing button.

- **Tab: Data**

The Data panel consists of a list of reports, which the tool should provide based on the generated tree. (See 7.1).

- **Tab: Customize**

This panel allows the user to customize the nodes of the tree. It includes an `<input>` tag with a button to change the text of the selected node. The panel also includes controls for changing the color and shape of the node. The shape can be changed by choosing between squares and circles for the shape, and choosing between black, red and green for the colors. To create a child as a defense node, the user can change the node's connector to be a connected line or a dotted line. Finally, two options for AND/OR refiners are added to switch between them.

The “Other Tools” menu consists of three buttons: **Download**, **Print**, and **Support**. The **Download** button enables users to download the tree, the **Print** button opens a print window option, and the **Support** button redirects users to the documentation page. However, the final section, which

has the largest part of the page in Figure 8, contains the box where the tree is displayed ③, and a simple toolbar for providing quick access to the basic functions ④, illustrated in Figure 11.



Figure 11: Toolbar menu

The menu in Figure 11 consists of six buttons, two buttons to zoom the tree in and out, a button to update the tree box, a button to delete the tree and two buttons to upload and download the tree file. The download button (Figure 12) enables the user to download the tree diagram as a .png or .jpeg file. Likewise, the upload button (Figure 13) allows the user to upload and generate a tree from an XML or JSON file. In addition, the section has a text box on the left side of the toolbar for naming the diagram. The user can rename the tree diagram directly by entering a new name in this box, which will be used in the downloaded file.

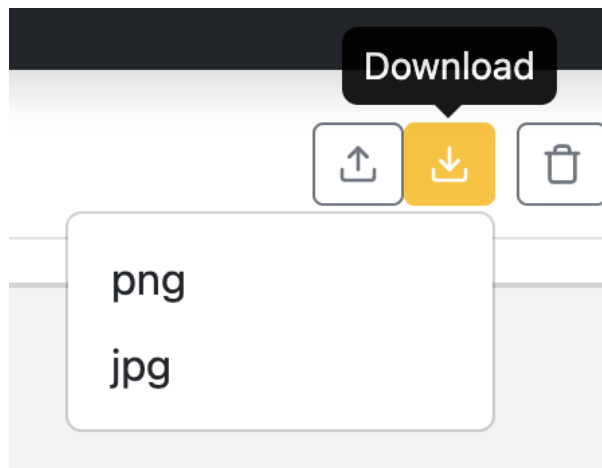


Figure 12: Download button

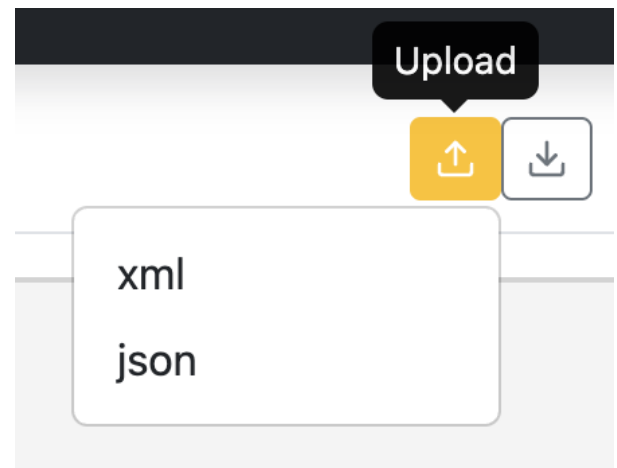


Figure 13: Upload button

In the Toolbar buttons, the Tooltip feature is used to ensure the functions of the buttons are unambiguous for the users. This feature provides a simple informative message box when the user hovers over the related elements, as shown in Figure 12.

Javascript for Tooltip feature:

```
document.querySelectorAll('[data-bs-toggle="tooltip"]')
    .forEach(tooltip => {
        new bootstrap.Tooltip(tooltip)
    })
```

HTML:

```
<span data-bs-toggle="tooltip" data-bs-title="Reset">
    <button type="button" class="btn" id="refreshBtn" onClick="history.go(0);">
        Refresh</button>
</span>
```

4.3 Heuristic Evaluation

The *Usability Heuristic Evaluation* is a set of ten qualitative guidelines developed by Nielsen [39]. The method involves analyzing applications according to the ten rules, then finding usability problems in the design, and finally evaluating the problems by assigning numbers from 0 to 4 [35], as shown below:

- 0 = I don't agree that this is a usability problem at all
- 1 = Cosmetic problem only: need not be fixed unless extra time is available on project
- 2 = Minor usability problem: fixing this should be given low priority
- 3 = Major usability problem: important to fix, so should be given high priority
- 4 = Usability catastrophe: imperative to fix this before product can be released

After developing the ADT web application based on the HCI principles discussed in section 4.1.3, we analyze the actions performed by the user according to Nielsen's heuristic evaluation rules [39]. These rules were summarized, simplified and structured by Dix [35] as follows:

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose and recover from errors
10. Help and documentation

These usability heuristic evaluation rules closely align with the principles of HCI (see section 4.1.3), facilitating the successful implementation and adherence to the essential design rules and principles within the interface. In collaboration with my supervisor, Mr. Schiele, we conducted an evaluation of the interface, by systematically examining all functionalities and assessing their compliance with the heuristic evaluation rules. The interfaces of the implemented web application align with the majority of usability evaluation rules and studied HCI principles, as presented in 4.1.3 and 4.2.2. However, during the evaluation, a few usability points were identified where improvements are necessary. These issues are as follows:

1. **Issue (1) : The controls in the Child tab do not have similar usability methods and lack consistency.**

- **Violated Rule:** Consistency and standards
- **Rating:** 3
- **Description:** In the first version of the ADT interface, the functions of add child and remove child were different, as in Figure 14. This difference makes it difficult for the user to easily understand how to update a child node.
- **Solution:** To enhance the user's understanding of the usability of this feature, the forms for adding and removing child nodes have been updated to be consistent, as in Figure 15.

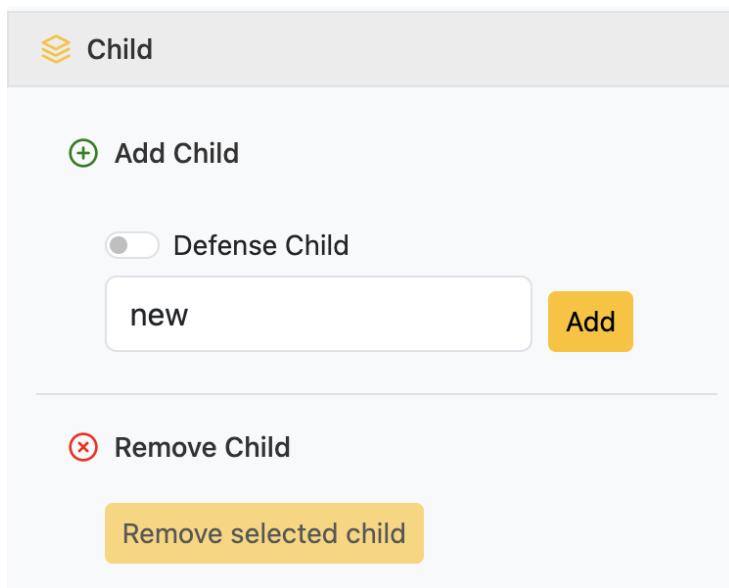


Figure 14: An inconsistent add-and-remove child design and methods.

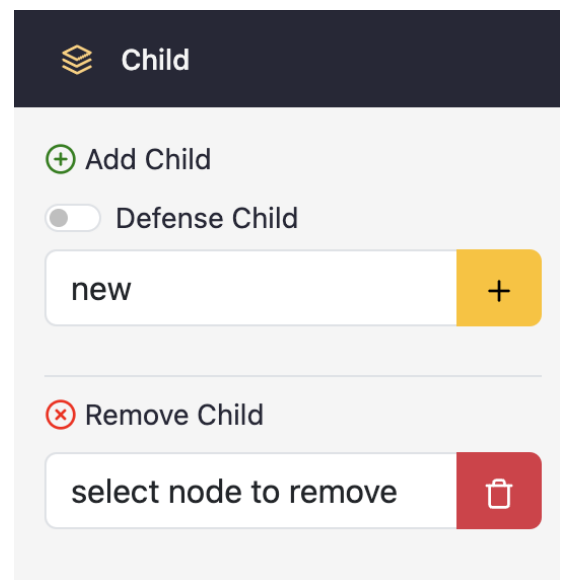


Figure 15: Consistent add-and-remove child methods and design.

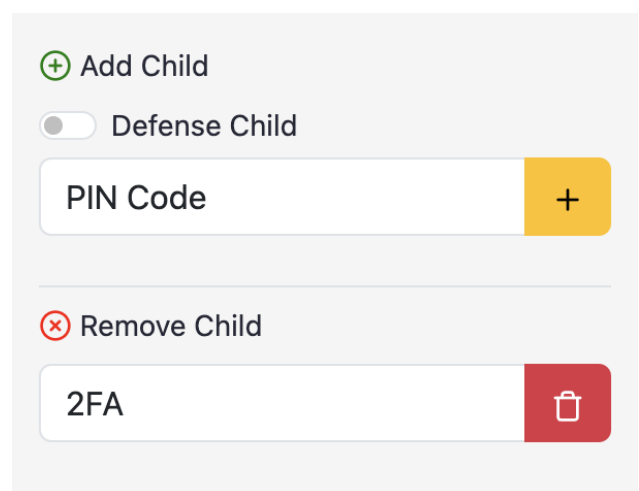
2. Issue (2) : Users can submit a tree or new child node with empty text.

- **Violated Rule:** Error prevention
- **Rating:** 3
- **Description:** The submission buttons can be pressed in Create and Child tabs when the user does not add an input in the related text-box. This will result in an empty submission request, which will force the user to perform more actions to fix it.
- **Solution:** To prevent the user from making this mistake, the buttons remain disabled until the related text-boxes have input, as demonstrated in Figures 16 and 17.



The screenshot shows a form with two sections. The top section is titled "Add Child" with a green plus icon and a "Defense Child" toggle switch. Below it is a text input field containing "text.." and a yellow button with a "+" sign. The bottom section is titled "Remove Child" with a red X icon and a text input field containing "text.." and a red button with a trash icon. Both buttons are disabled.

Figure 16: Buttons are disabled before inserted input.



The screenshot shows the same form as Figure 16, but with the text input fields filled with "PIN Code" and "2FA". The yellow "+" button and the red trash icon button are now enabled and highlighted.

Figure 17: Buttons are enabled after the input is inserted.

3. Issue (3) : Users are unable to determine the status after submitting the tree text.

- **Violated Rule:** Help users recognize, diagnose and recover from errors
- **Rating:** 2
- **Description:** After form submission, the user is not able to see whether the submission process went successfully or failed.
- **Solution:** To prevent the user from making this mistake, a status message should be displayed to the user after submission, as in Figures 18 and 19.

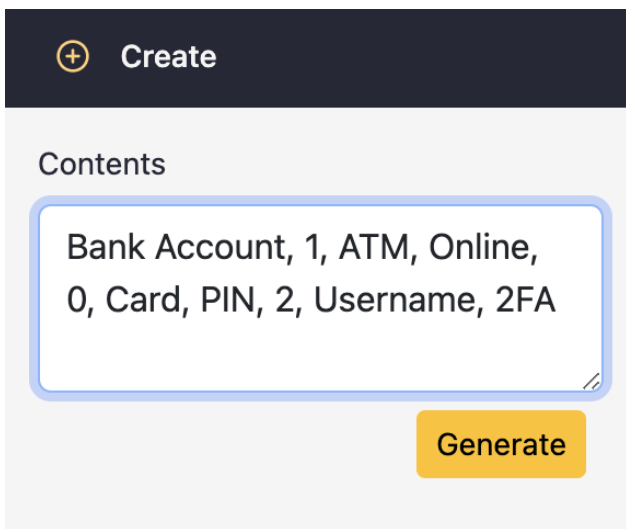


Figure 18: User is ready to submit the contents of the text box.

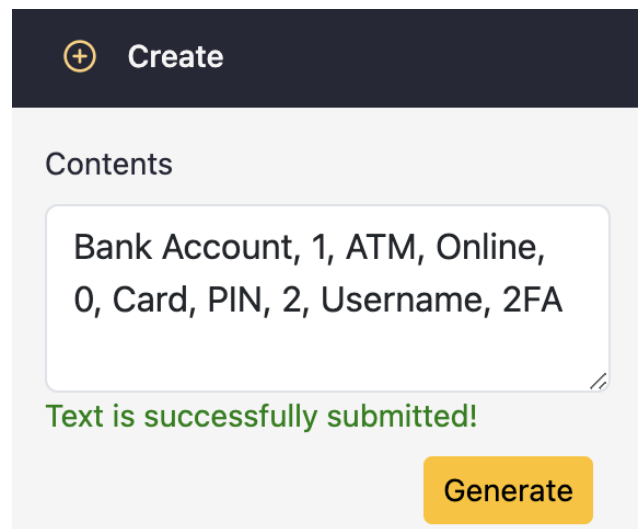


Figure 19: Status after the user has pressed the Generate button.

4. Issue (4) : The symbols of the pages in the navigation bar are not clear.

- **Violated Rule:** Visibility of system status
- **Rating:** 2
- **Description:** In the first version of the ADT interface, the navigation bar had only two buttons to switch to the home page and the documentation page. However, the buttons were designed as symbols that indicate the pages instead of simple text. This can be seen in Figure 20. Besides, the current page (ADT) was not displayed in the navbar, which does not help the user to know where he is in the web application.
- **Solution:** In order to provide users with a clear overview of the web application pages, the symbols have been replaced with simple text for the three pages. Additionally, the current page name is highlighted, as shown in Figure 21.



Figure 20: The first design of the navigation bar of the ADT web application.



Figure 21: The final design of the navigation bar of the ADT web application.

5. Issue (5) : Users may find it confusing that the links in the Data tab appear to be clickable but are not actually functional.

- **Violated Rule:** Visibility of system status
- **Rating:** 2
- **Description:** The functionality within the Data tab (Figure 22) has not been implemented yet. These options are planned for future development (see 7.1). Consequently, their presence may confuse users, leading them to click on the options and not receive any expected results.
- **Solution:** To ensure users have a clear overview of the available features within the tab, the labels of the features have been disabled and are displayed -as in Figure 23- with a light gray color to resolve the confusion.

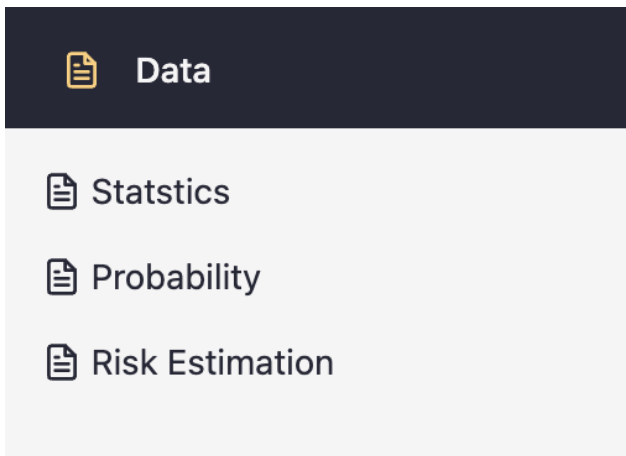


Figure 22: Data tab contents before design changes.

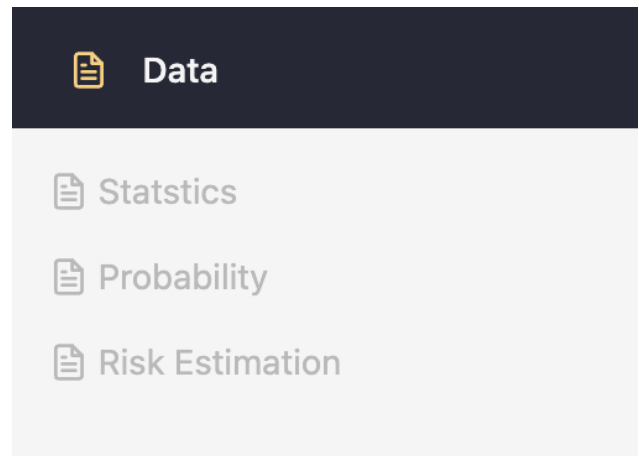


Figure 23: Data tab contents after design changes

5 Results

In this chapter, the second objective (2) of the study is fulfilled by presenting the final results of the developed web application interfaces.

5.1 Interfaces

The following figures show the final design of the interfaces of the ADT web application. This includes the home page, the documentation page, the ADT page and the contents of the sidebar tabs.



Figure 24: The ADT tool page

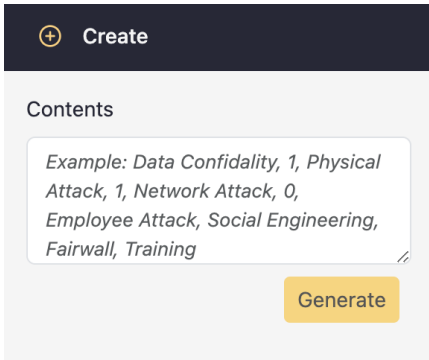


Figure 25: Create tab status of an empty input.

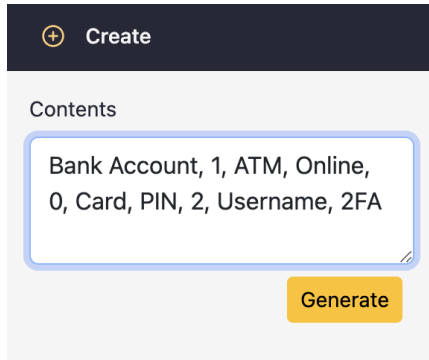


Figure 26: Create tab status of an entered input.

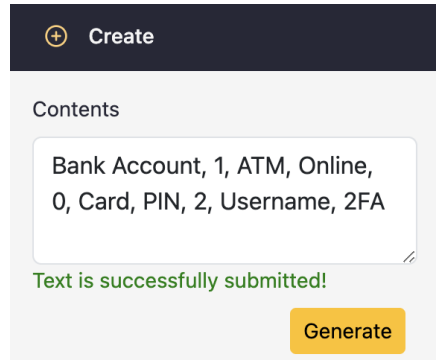


Figure 27: Create tab status after input submission.

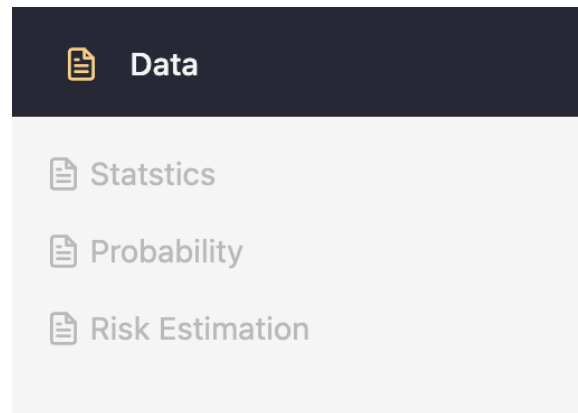
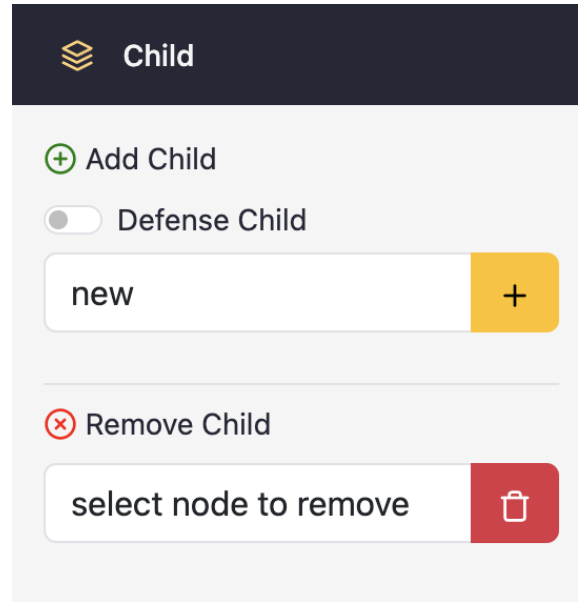
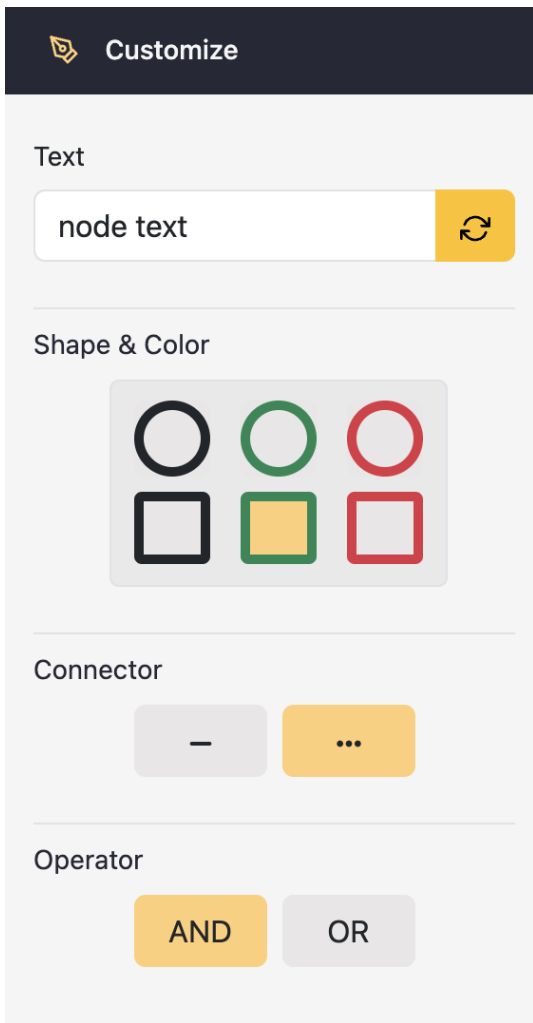


Figure 28: ADT menu tabs

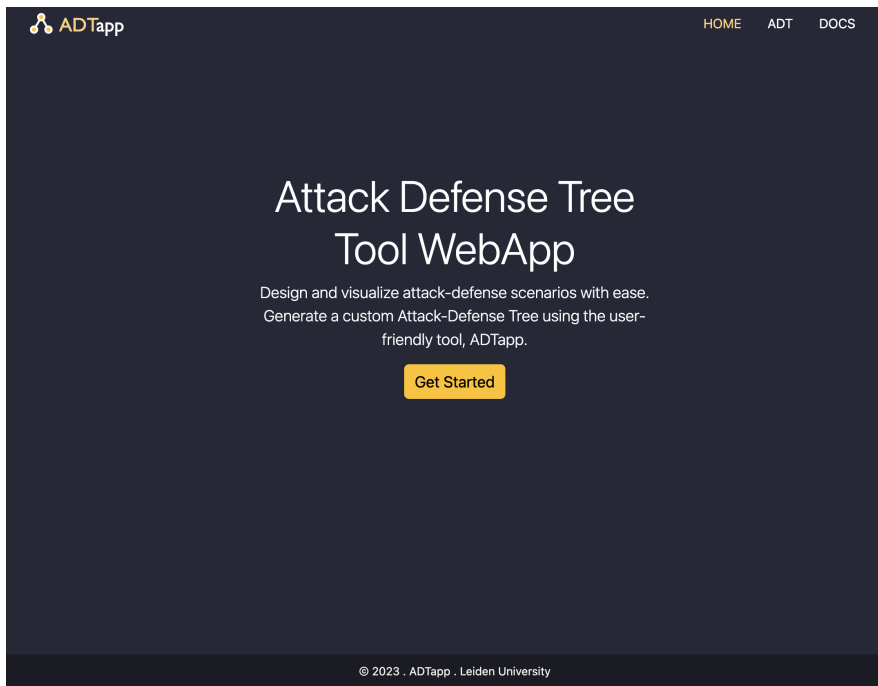


Figure 29: Home page

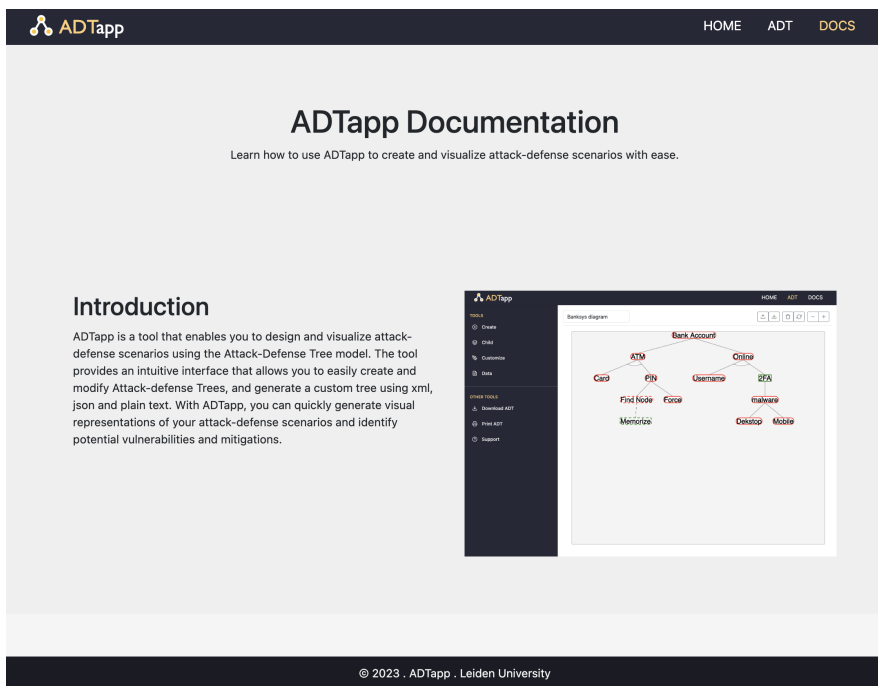


Figure 30: Docs page

6 Evaluation

6.1 Comparison

There have been several valuable tools developed to construct ADTs that offer beneficial functionality for evaluating attack scenarios. One of the well-known tools in this area is ADTool [12]. In this section, the application of the current research will be compared with the ADTool software. This comparison is executed individually by myself, based on the previously conducted evaluation.

*	ADTool		Current study's ADT WebApp	
Criteria	Evaluation	Description	Evaluation	Description
Consistency	Good	The design of the elements in the tool is consistent.	Good	The design of the elements in the web-page is consistent.
Simplicity	Needs Improvement	The user needs to perform multiple actions every time the tree needs modification.	Good	The user interacts with one simple menu that consists of all required control.
Learnability	Needs Improvement	It needs a certain level of technical expertise to use it and interpret the results.	Good	No expertise is needed. The interface has a simple control method.
Documentation	Good	Detailed documentation is provided by the tool.	Fair	Adequate documentation is provided by the tool.
Efficiency	Needs Improvement	The transition within the user control is not seamless and the user should move between multiple windows.	Good	Seamless transition within the user controls, where all controls are provided in the same place based on the function.
Feedback	Fair	Message log display messages about the current status of the tree.	Fair	Helpful error messages are provided.
Error tolerance	Fair	Alter and remove node options are provided.	Fair	Submit buttons are restricted by conditions to prevent wrong input.
Mapping	Good	Users can easily understand the relationship between the visual elements and the actions performed.	Good	Users can easily understand the relationship between the visual elements and the actions performed.

Table 1: A comparison between ADTool and ADT web application based on usability principles.

Table 1 presents a comparative analysis between the ADTool software and the ADT web application of the current study, with a particular focus on the objective characteristics of the usability principles. However, it is important to consider that due to certain limitations (see 6.2), the ratings presented in Table 1 are subject to potential subjectivity, as they rely solely on my evaluation. Furthermore, the evaluation of other usability elements, particularly, the visibility principles, relies more on the feedback that the user provides after testing the application. Consequently, these principles are not presented in Table 1 because the evaluation may vary from one user to another. Therefore, to ensure a comprehensive and reliable assessment, a user evaluation should be conducted for the provided comparison (see 7.1).

6.2 Limitations

Due to certain limitations, several aspects have not been discussed or fully addressed in this research. More specifically, this study is the front-end implementation of the ADT web application project, while the broader project encompasses various sub-research areas, such as back-end development, plain text to XML conversion, ADT visualization, tree generation and others. Considering that the implementation and the planning of these aspects are still in progress, and that the allocated time for the current study is limited, it has been challenging to accomplish certain tasks. Particularly, the tasks related to implementing the back-end functionalities. Therefore, the remaining work requires further implementation in the future (see 7.1).

Furthermore, because of the unfortunate limitation of lacking tree visualization in the current implementation, it was challenging to evaluate and make improvements based on user feedback. As a result, the assessment provided by me in this study (1) can be subjective. Therefore, once the application is fully developed, it is necessary to conduct an evaluation through user participation [40] in order to enhance the usability of the ADT web application accordingly.

7 Conclusion

The ADT application is designed as a web-based platform, which enables users to access it across various devices without the need for downloading or installation. The research findings indicate that users can create trees within the application through multiple approaches, including uploading XML or JSON files, as well as inserting plain text using natural language. Upon generating the tree, users have the ability to adjust its structure by adding (defense) child nodes to a selected parent node and customizing them according to their preferences. The interfaces provided offer straightforward interactions with the tree, devoid of complex intermediate functions or activities. Consequently, it is anticipated that users will find the application easy to learn, and the integration of combined HCI principles will contribute to achieving a high level of user-friendliness and satisfaction, which is the main purpose of this study.

To conclude, it is essential to ensure that the utilization of the ADT technology is easily accessible to users, by enabling them to evaluate security systems with it without requiring extensive technical expertise. To accomplish this, two fundamental objectives were set in the research: Objective 1 (**O1**) is to identify the main design concepts that improve the usability of web applications, and Objective 2 (**O2**) is to build a web application for ADT with the studied design concepts in O1 applied on it.

Since interface design plays a significant role in the usability of the applications, the research has fulfilled **O1** by exploring in depth the principles of HCI and how they are applied to the interface of web applications. Secondly, the research satisfied **O2** by employing the principles and the characteristics of visibility, usability, and interaction to design a dedicated web-based application for constructing ADTs. This approach aimed to ensure that the tool is effectively accessible to users, considering their determined requirements and varying capabilities. After implementing the first design of the web application, a usability heuristic evaluation is applied on it to check the design decisions developed and modify them based on the issues found. Finally, the design of the resulting web application was compared to the design of a previous application in the same domain. This comparative analysis aimed to assess the effectiveness and the impact of the updates and decisions made to the web application design.

7.1 Future Work

The web application still requires to have controls for analyzing the attacks and adding costs for each tree node. This can be accomplished by implementing an input box and a corresponding label in each node, allowing users to input costs for the nodes. Moreover, the data tab (Figure 28) demands further work to perform calculation functions, such as generating reports and automatically computing the probabilities and the costs of the nodes based on the given data. Therefore, more controls within the tab should be designed based on the report's requirements.

According to the discussed limitations (6.2) associated with the visualization of the tree, the ADT web application still requires a user evaluation in further studies. This can be accomplished by observing users as they interact with the application and documenting the challenges and errors they encounter during the testing process. An alternative approach could involve creating a survey containing questions on usability principles related to the interface. This would enable us to identify the features that require further improvement.

However, since the project is not complete and does not contain a database yet, the interfaces of the web application are developed with plain front-end technologies. Nowadays, the majority of online applications leverage cutting-edge front-end framework technologies to optimize website performance. Therefore, it is important to adopt one of these frameworks, such as `React.js` or `Angular.js`, during the back-end implementation phase.

Finally, the current home page of the web application has a simple text to introduce the tool. However, the purpose of this page is to provide more details about the services the ADT web application provides, including the features of the tool, the technologies used, the team contributed, and ADT specifications. These information should be included when the work is complete and all the services and functions have been built. The aforementioned is applicable to the documentation page as well.

References

- [1] IBM Security. Cost of a data breach. report 2022. *Ponemon Institute*, <https://www.ibm.com/downloads/cas/3R8N1DZJ>, 2022.
- [2] B. Schneier. Attack trees. *Dr. Dobb's Journal of Software Tools*, 24:21–29, 1999.
- [3] D. Katsabas, S. M. Furnell, and A. D. Phippen. It security: A human computer interaction perspective. *Advances in Network Communication Engineering*, 2:35–42, 2005.
- [4] S. L. Polasanapalli and P. Buggareddy. Usability evaluation to design a user interface by implementing hci design principles. *Bachelor Thesis in Computer Science*, Blekinge Institute of Technology, 2020.
- [5] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer. Foundations of attack–defense trees. *Lecture Notes in Computer Science*, page 80–95, 2010.
- [6] B. Shneiderman. Designing the user interface. *International Journal of Computer Science*, 1998.
- [7] D. Norman. The design of everyday things. *Revised and Expanded Edition*, pages 1–247, 2013.
- [8] Ch. Gong, Y. Qiu, and B. Zhao. Establishment of design strategies and design models of human computer interaction interface based on user experience. *Beijing Institute of Technology*, 2018.
- [9] F. Delcourt and T. Weiser. Graphical user interface for dag-based attack trees. *Ecole polytechnique de Louvain, Université catholique de Louvain*, 2022.
- [10] A. Opel. Design and implementation of a support tool for attack trees. *Department of Mathematics and Computer Science. Technical University Eindhoven*, 2005.
- [11] Adtool. *Security and Trust of Software Systems*, <https://satoss.uni.lu/members/piotr/adtool/>.
- [12] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer. Adtool: Security analysis with attack–defense trees. *University of Luxembourg*, Extended Version:173–176, 2013.
- [13] O. Gadyatskaya, R. Jhavar, P. Kordy, K. Lounis, S. Mauw, and R. Trujillo-Rasua. Attack trees for practical security assessment: Ranking of attack scenarios with adtool 2.0. *Quantitative Evaluation of Systems. Lecture Notes in Computer Science*, 9826:159–162, 2016.
- [14] B. Kordy, P. Kordy, and Y.V.D Boom. Sptool – equivalence checker for sand attack trees. *Risks and Security of Internet and Systems. Lecture Notes in Computer Science*, 10158:105–113, 2017.
- [15] E. Ruijters and M. Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer science review*, pages 15–16, 29–62, 2015.

- [16] S. Pinchinat, M. Acher, and D. Vojtisek. Atsyra: An integrated environment for synthesizing attack trees. *Graphical Models for Security. Lecture Notes in Computer Science*, 9390:97–101, 2016.
- [17] R. Kumar, S. Schivo, E. Ruijters, B.M. Yildiz, D. Huistra, J. Brandt, A. Rensink, and M. Stoelinga. Effective analysis of attack trees: A model-driven approach. *Formal Methods and Tools, University of Twente*, 2018.
- [18] H. S. Lallie, K. Debattista, and J. Bal. A review of attack graph and attack tree visual syntax in cyber security. *Computer Science Review*, 35:100219, 2020.
- [19] A. Yathuvaran and D. Leveille. (at-at) attack tree analysis tool. <https://github.com/yathuvaran/AT-AT>, 2021.
- [20] M. Eckhart, K. Meixner, D. Winkler, and A. Ekelhart. (adtgenerator) securing the testing process for industrial automation software. *Computers Security*, <https://github.com/sbaresearch/adtgenerator>:156–180, 2019.
- [21] Softeam RD Department. Attack tree designer, modelio. *European Commission. CPSwarm H2020 project*, <https://github.com/Modelio-R-D/AttackTreeDesigner>, <https://github.com/cpswarm/modelio-attack-tree-module>, 2020.
- [22] J. B. Hong, D. S. Kim, Ch. Chung, and D. Huang. A survey on the usability and practical applications of graphical security models. *Computer Science Review*, 26:1–16, 2017.
- [23] K. Edge. A framework for analyzing and mitigating the vulnerabilities of complex systems via attack and protection trees. *Ph.D. Thesis, Air Force Institute of Technology, Wright Patterson AFB, AAI3305523*, 2007.
- [24] M. Dacier and Y. Deswarte. Privilege graph: An extension to the typed access matrix model. *Lecture Notes in Computer Science*, 875:319–334, 1994.
- [25] X. Ou, W. Boyer, and M. McQueen. A scalable approach to attack graph generation. *Proc. of the 13th ACM Conference on Computer and Communications Security*, page 336–345, 2006.
- [26] K. Ingols, R. Lippmann, and K. Piwowarski. Practical attack graph generation for network defense. *Proc. of the 22nd Annual Computer Security Applications Conference*, page 121–130, 2006.
- [27] Y. Liu and H. Man. Network vulnerability assessment using bayesian networks. *B. V. Dasarathy (Ed.), Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2005, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 5812:61–71, 2005.
- [28] M. McQueen, W. Boyer, M. Flynn, and G. Beitel. Quantitative cyber risk reduction estimation methodology for a small scada control system. *Proc. of the 39th Annual Hawaii International Conference on System Science*, 9, 2006.
- [29] S. Jajodia, S. Noel, and B. OBerry. Topological analysis of network attack vulnerability. *Managing Cyber Threats, in: Massive Computing*, 5:247–266, 2005.

- [30] N. Tippenhauer, W. Temple, A. Hoa Vu, D. Nicol B. Chen, Z. Kalbarczyk, and W. Sanders. Automatic generation of security argument graphs. *Proc. of the 20th IEEE Pacific Rim International Symposium on Dependable Computing*, page 33–42, 2014.
- [31] D. Baca and K. Petersen. Prioritizing countermeasures through the countermeasure method for software security (cm-sec). *Product-Focused Software Process Improvement, in: Lecture Notes in Computer Science*, page 176–190, 2010.
- [32] J. Hong and D. Kim. Hierarchical attack representation models for network security analysis. *Proc. of the 10th Australian Information Security Management Conference on SECAU Security Congress*, page 74–81, 2012.
- [33] M. B. V. Kandababu and S. G. V. Indukuri. Analysis on website design using usability principles. *University of Borås, School of Business and IT*, 2010.
- [34] C. Adhitya, R. Andreswari, and P.F. Alam. Analysis and design of ui and ux web-based application in maiprojek startup using user centered design method in information system program of telkom university. *IOP Conference Series: Materials Science and Engineering*, 1077 012039, 2021.
- [35] A. Dix, J. Finlay, Gregory, D. Abowd, and R. Beale. Human-computer interaction. *Pearson Education*, 3rd Edition, 2003. Ch 9.3.
- [36] Y. Rogers, H. Sharp, and J. Preece. Interaction design: Beyond human - computer interaction. *John Wiley Sons*, 2011.
- [37] I. Zubrycki, M. Kolesinski, and G. Granosik. Graphical programming interface for enabling non-technical professionals to program robots and internet-of-things devices. *Advances in Computational Intelligence. Lecture Notes in Computer Science*, 10306:620–631, 2017.
- [38] V. Hinze-Hoare. Review and analysis of human computer interaction (hci) abstract principles. *CoRR*, abs/0707.3638, 2007.
- [39] J. Nielsen and L. Mack. Usability inspection methods. *John Wiley and Sons, New York*, pages 25–64, 1994b.
- [40] A. Dix, J. Finlay, Gregory, D. Abowd, and R. Beale. Human-computer interaction. *Pearson Education*, 3rd Edition, 2003. Ch 9.4.